Welcome to SI539
Building Complex Web Sites
Ruby on Rails

Charles Severance

What is the Course About?

- Learning about HTML, CSS - designing a web site's look and feel
- Developing software driven web sites - using Ruby on Rails
- Storing data in data

- ====>> Design of Complex Web Sites

Overall Course Goal

- Expose you to a wide range of material in a hurry
- Equivalent to 4-5 semesters computer science courses
- You won't be an expert in this stuff ...

- .... But you will be very dangerous to yourself and others

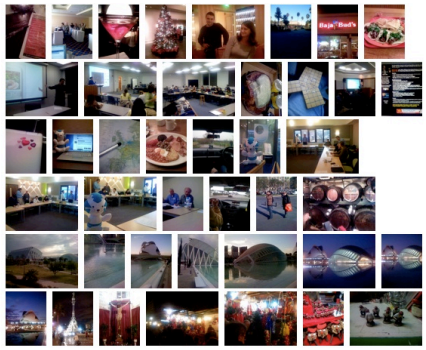
SI539 over the years..

- SI539 has a tradition of using the coolest and hippest web programming language
  - Cold Fusion
  - PHP
  - Ruby on Rails

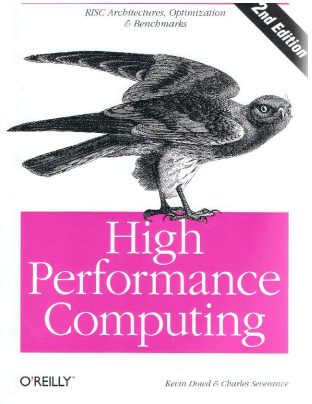
Syllabus

Dr. Charles Severance
Clinical Assistant Professor
Office Hours: by appointment
Email: csev@umich.edu

www.dr-chuck.com
www.dr-chuck.com/csev-blog
twitter.com/drchuck/
www.dr-chuck.com/media.php
www.dr-chuck.com/images/



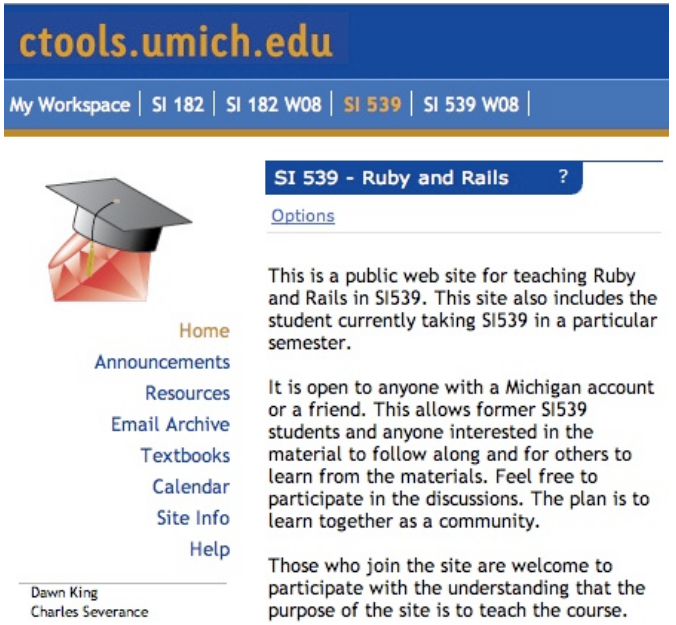
- My previous job: Sakai / CTools Architect
- My research topics: Software For Teaching and Learning, Web Lecture technologies, and High Performance Computing.
- I also work in developing standards for learning software interoperability
- Hobbies: Hockey, Off-Road Motorcycle Riding



Sakai

Course Site

- Two sites
  - Semi-public - auditors and helpers
  - Private - Primarily grading
- Mailing list
  - si539@ctools.umich.edu
  - Please use it like a conversation



Textbook - 1

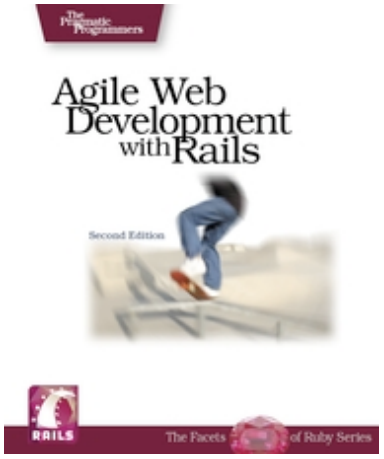
- Build Your Own Web Site the Right Way Using HTML & CSS by Ian Lloyd
- Published: Sitepoint, 2006
- ISBN: 0975240293; 9780975240298
- web site: http://www.sitepoint.com/books/html1/

Textbook - II

- Build Your Own Ruby on Rails Web Applications by Patrick Lenz
- Published: Sitepoint, 2007
- ISBN: 0975841955; 9780975841952
- web site: http://www.sitepoint.com/books/rails1/

Textbook - III

- Agile Web Development with Rails (2nd ed) by Dave Thomas and David Heinemeier Hansson
- Published: Pragmatic Bookshelf, 2007
- ISBN: 0977616630; 9780977616633
- web site: http://pragprog.com/titles/rails2/



What is Ruby?

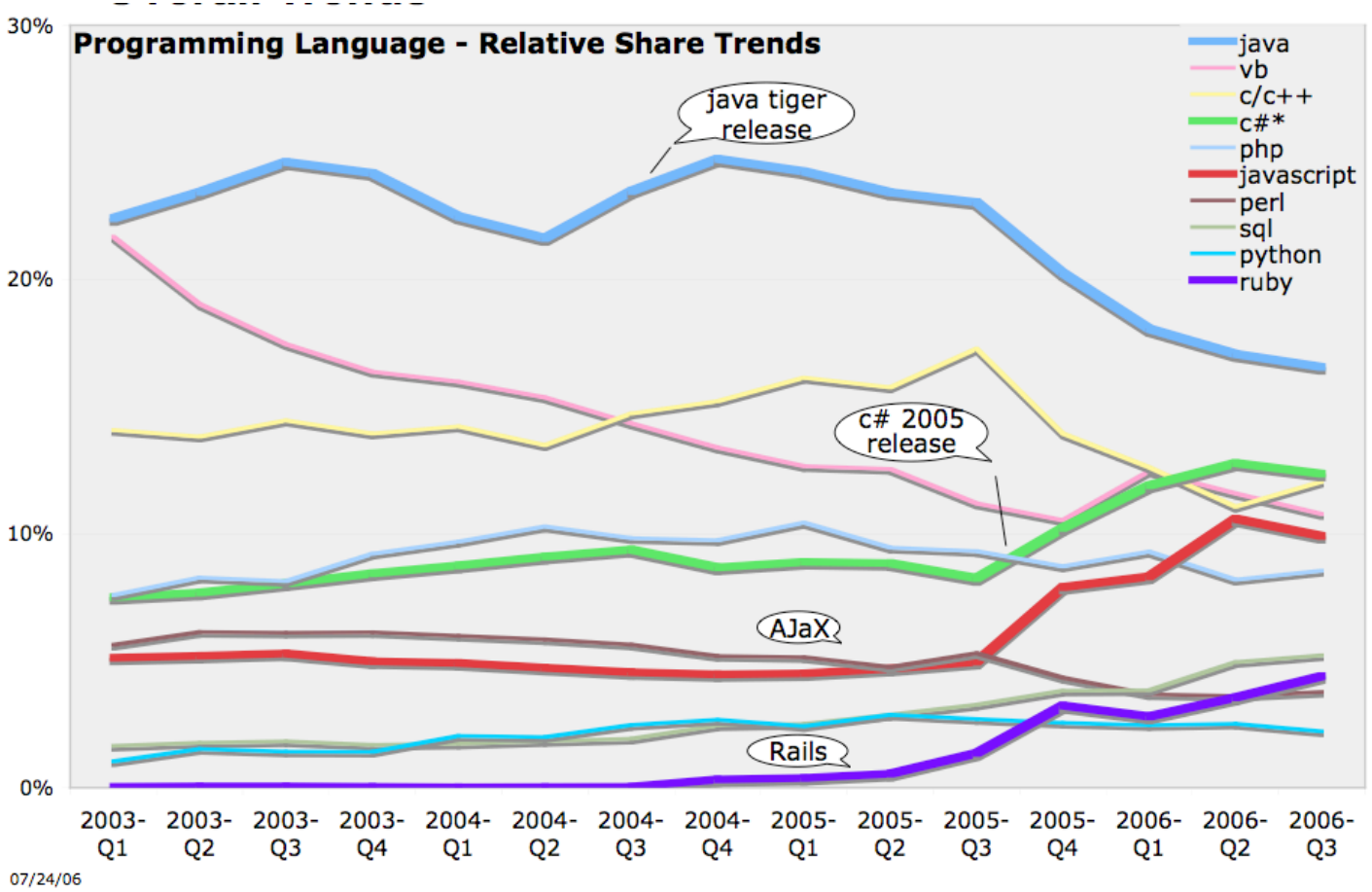
- Ruby is a programming language
  - At one level any programming language can be used to solve any problem - everything is *possible* in any language
  - Languages differentiate them selves based of ease of use, elegance, power, simplicity, efficiency, and many other subtle but very important factors

Ruby's Advantages

- Truly Object Oriented from the ground up - very consistent patterns to the language
- User-friendly language syntax - easy to read and easy to learn
- Help the user develop nice, clean, powerful programs
- Very powerful run time class extensions - make applications like Rails not only possible but very natural

Ruby History

- Originally developed in Japan by Yukihiro Matsumoto (aka “Matz”)
  - “I wanted a scripting language that was more powerful than Perl, and more object-oriented than Python. That's why I decided to design my own language.”
- Work started in 1993, and released it to the public in 1995.
- "Ruby" was named as a gemstone alluding to name of Perl
- Also pearl is the birthstone for June, and ruby is the birthstone for July.
- English version in 1998



http://radar.oreilly.com/archives/2006/08/programming_language_trends_1.html

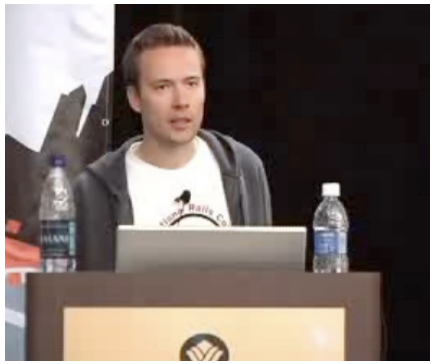
What is Rails?

- Rails is a full-stack web application framework
  - The Rails framework includes common functionality that is useful across a wide range of web applications
  - By starting with Rails - you get a lot for "free" - you can focus on your application and its data rather than writing lots of plumbing
- Rails is written in Ruby

http://www.youtube.com/watch?v=PQbuyKUaKFo

Rails History

- David Heinemeir Hansson from Denmark developed a web application called Basecamp
- Rails was created by taking the general purpose bits of the Basecamp application out so they could stand on their own
- Rails 1.0 was released in December of 2005.



http://www.youtube.com/watch?v=mp4z2eK1Avw

Helping Others

- Please ask for and/or give help
- In the beginning this is very foggy - hard to find the big picture
- But remember that your purpose is to learn
- Ask the mailing list - post code bits - it is OK

http://en.wikipedia.org/wiki/Blind_Men_and_an_Elephant

Chuck's Basic Rules

- Coming late or leaving early - OK
- Sleeping in class - OK
- Using a laptop - OK
- Eating or drinking - OK if the room permits it
- Stepping out to take a bio break - OK
- Asking questions at any time - OK
- Correcting me when I make a mistake - OK
- Skipping class - not very wise - but OK
- Doing things that distract other students or making difficult for us all to learn - Not OK
- Skipping class or sleeping in class and then expecting me to repeat entire lectures in office hours - Not OK
- Waiting to the last minute and asking me to review the whole semester in office hours - Not OK

Programming Assignments

- Probably the most important part of the course - our task is to teach you programming skill - so you can ultimately do this on your own
- Increasing difficulty over time
- I monitor areas of difficulty and adjust the course material, lectures, assignment difficulty, everything
- Sometimes it helps to do the assignments twice if you are struggling
- Must be handed in on time - so I can distribute sample solutions

Exams

- Two practical exams
  - Come to Lab - I hand out a programming problem - must finish and hand in within 2 hours - open book, open notes, open laptop, can look at your old programming assignments, surf the web - just no help from other people.
- Two written exams - classic stuff on paper
  - Short answer, multiple choice, read code and tell what it does, very little code writing - page of notes

Participation

- In class-exercises - handed out - collected - not formally graded
- In class questions / comments
- Helping other students
- Giving me feedback about the course
- ...

Grading

- Approximate percentages
  - Assignments: 50% Exams: 40% Participation: 10%
- Straight scale from written syllabus

Course Outline

- HTML and CSS - Web site design
- Running code in the server
- Database Design
- Advanced Topics

Project Option

- After the midterm I have a good indication of how well you can learn the material
- Strong students have the option to do a project
- Can be a single individual or group - you must make a proposal to me and I must approve it.
- Project groups must file weekly reports - progress must happen every week to get a grade.

Project Option

- Students doing projects to not have to do weekly assignments - project reports substitute for the assignments
- Students doing projects *are* responsible for all the lecture and book material - and must take all of the exams.
- Project teams will make a report, presentation, and portfolio web site at the end of the semester

Success in The Course

- Don't wait until the last minute each week
- If you get stuck on something - move around - review some material - read the book - then come back
- When you look back - you will see that this was all *really* easy
- When you feel stuck - communicate - use the list - ask a friend
- I need to get feedback - a lot

Beware of Overconfidence

- Students who have some prior experience may be at some disadvantage because the class may seem to easy and/or too slow.
- Start to skip lectures and labs - just do the assignments by themselves.
- Once the course starts to speed up - they get lost quickly and find themselves a few weeks behind.
- Solution: Come to class and lecture and catch up on E-Mail with one ear on the material. Also help begining students to make sure *you* understand.

No Experience Required

- I am committed to teaching the course to students with no prior experience in programming.
- I will alter the pace and/or order of the material as I see a need based on how well students are doing.
- Make sure to let me know on the mailing list, or by private mail or talking to be in lecture or lab how you think we are doing - or if you missed something.

Welcome to the course...

- Any questions?